COST 526
“Automatic Process Optimization in Materials Technology” (APOMAT)

Half-Yearly Report

To be sent to **V.Tesch@access.rwth-aachen.de** until **February 28, 2003**

1. Reporting Period	**1.7.2002 – 31.12.2002**
Project title	Optimization of Fatigue Resistance of Cold Forging Tools by Considering Damage Mechanisms at Micro Scale
Project leader	Dr. Igor Gresovnik
Organization	C3M
Main collaborators involved	Faculty of Natural Sciences and Technology, University of Ljubljana, Slovenia.

2. Funding Situation	
Amount of money received specifically for COST	2.9 kEuros
Other resources partially used for the project	kEuros

3. International Collaboration
(mention group and type of work done in collaboration during the reporting period)

Participation in the Working Group Meeting in Budapest + project progress report

❐ YES ➽

❐ NO

IPPT Warsaw; Contact sensitivities and preliminary overview of smooth contact formulations.
Rockfield Software, Swansea; definition of interfacing utilities between the optimisation shell and the simulation programme “Elfen”.

4. Industry participation
(mention name of companies and work done in collaboration during the whole project)

Iskra-Avtoelektrika. Analysis of service life of tooling systems

5. Meetings, visits, exchange of scientists, short-term scientific missions	**Location, date**
Working Group Meeting	Budapest, Nov 28/29, 2002
Meeting with Iskra-Avtoelektrika	Ljubljana, 1 meeting

6. Progress within the reporting period
(Not exceeding 3 pages, including tables and figures)

Work was focused on the development of the file interface between the optimisation shell *Inverse* and the finite element simulation environment *Elfen*. The interface enables the optimisation shell to manipulate the numerical analysis and run it in the optimisation scheme shown in Figure 1:

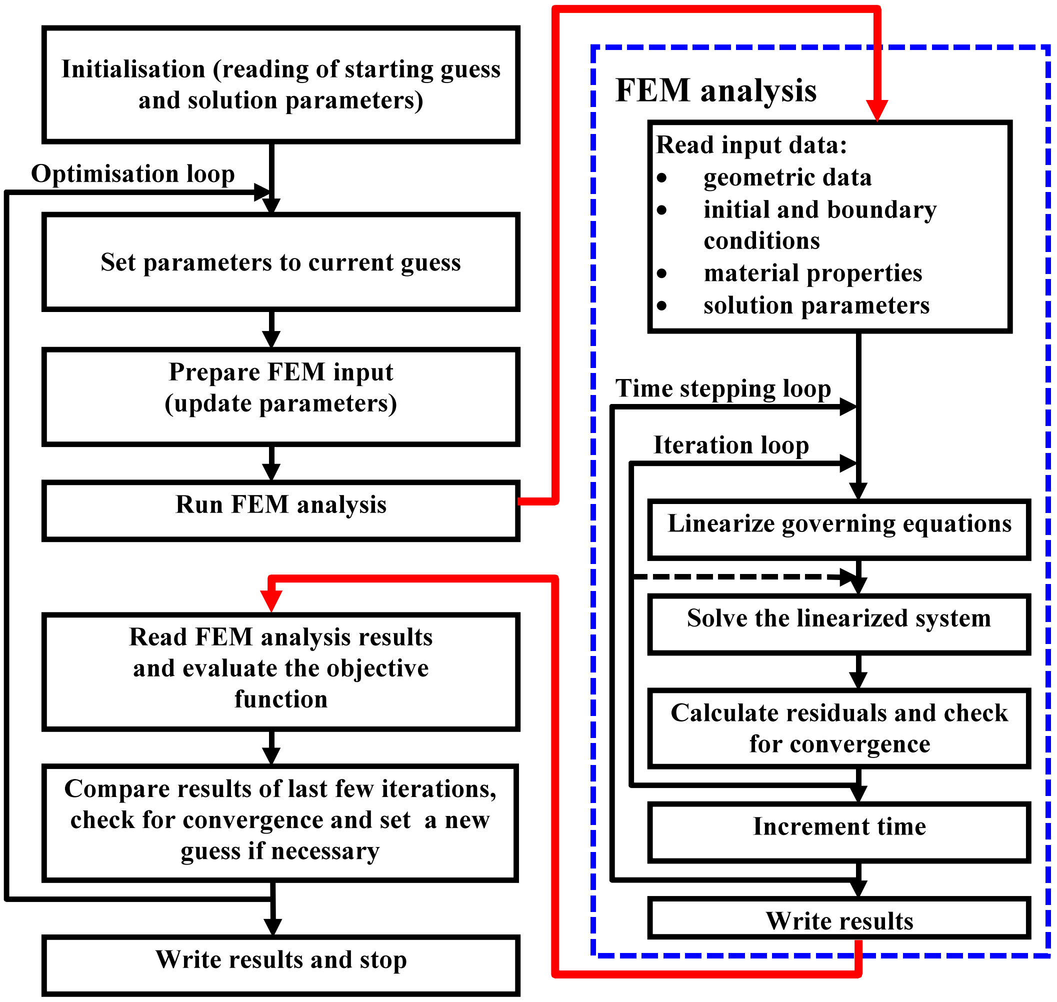
Initialisation (reading of starting guess and solution parameters)
Optimisation loop
Set parameters to current guess
Prepare FEM input (update parameters)
Run FEM analysis
Read FEM analysis results and evaluate the objective function
Compare results of last few iterations, check for convergence and set a new guess if necessary
Write results and stop
FEM analysis
Read input data:
• geometric data
• initial and boundary conditions
• material properties
• solution parameters
Time stepping loop
Iteration loop
Linearize governing equations
Solve the linearized system
Calculate residuals and check for convergence
Increment time
Write results

Figure 1: Function scheme of the optimisation shell “Inverse”

Outer part of the left-hand side of Figure 1 corresponds to the optimisation algorithm, which is built in the optimisation shell and is normally invoked through the corresponding command in the shell command file. In general, the optimisation algorithm requires a starting guess and some other input data such as convergence criteria. This data is passed to the algorithm through arguments of the call to the corresponding function in the shell command file. After initialisation steps, the algorithm enters a loop where it iteratively sets the optimisation parameters on the basis of pervious calculations, invokes the evaluation of the objective and constraint functionals, and checks for convergence. Evaluation of the relevant functionals is performed by another function of the shell shell. This function invokes interpretation of a specific *analysis block* of the command file, in which the user defines how quantities required by the algorithm are actually calculated. Among the others, this includes preparation of the input data for the finite element simulation according to the parameter values, running the programme that performs the analysis (right-hand side of Figure 1), reading the necessary results of the simulation and evaluation of the objective and constraint functionals on the basis of these results.

Some steps that are invoked in the *analysis block* of the command file are enabled by the functions of the interface. These include preparation of analysis input data that corresponds to the current values of optimisation parameters as set by the optimisation algorithm, running of the analysis and access to the necessary analysis results. Running of the analysis is enabled by the operating system while structure of the input/output interface depends mainly on the data representation rules of the simulation environment and other circumstances.

In the finite element environment Elfen, the analysis is performed by a single process, which reads input data such as mesh data, prescribed loads and displacements, contact data, loading data, etc., from one text file and prints all the results to another file. A skeleton for the input data can therefore be prepared in advance by the pre-processor, while the input interfacing utilities of the shell take care of modification of these data according to parameter values prior to each run of the analysis. Updating data in the input file according to parameter values is enabled by a special marking agreement, which enables to denote which pieces of data in the file correspond to which optimisation parameters. When the analysis is run for the first time, marks are searched for throughout of the input file and locations of data corresponding to optimisation parameters are stored in memory. At every subsequent iteration, these locations are used to determine which data in the analysis input file must be modified. This enables use of direct instead of sequential file access, which saves a lot of time when files are large.

Some optimisation parameters do not affect only elementary bits of data in the analysis input file. A typical example are shape parameters which can affect the whole finite element mesh. Update of the related data in the input file is in this case arranged differently. The mesh is read from the input file and stored in appropriate data structures of the shell, then the shape transforms are performed on this data, and finally the new finite element mesh is substituted back into the input file. In some cases, updates of geometry are performed by different steps which include running auxiliary external programmes.

What concerns reading of analysis results from the finite element analysis output files, the most concerning problems are related to large size of these files, which can easily reach tens of megabytes. Positions of specific data in the output file at a given optimisation iteration are usually not related to the positions of these data in previous iterations, which aggravates direct file access. Interface functions use different tricks to circumvent this problem. This includes taking advantage of formatting agreements whenever possible. For example, field data in Elfen output file is formatted in such a way that each component occupies the same amount of space regardless to its actual value, filling the unnecessary space by blanks. Dimensions of data fields are stated before the components are listed. This enables direct access of any component and jumping over the field if the beginning of the field and space occupied by a single component are known.

Another problem is related to the fact that data necessary for evaluation of objective and constraint functionals can be very dispersed. When user writes the code for reading analysis results from the output file, he or she does not want to bother with the question of the order in which data appear in the file, but wants to access any data randomly through descriptive interface functions. This is enabled by smart interface functions which store all information about the data in the result file, especially the positions, dimensions and formatting of fields encountered when searching for useful data, in a hierarchically organised database. Data collected in this way enable direct access to the information when the same or other interface function is called for the next time. Each time a given piece of data is needed from the output file, as many information as available from previous access is obtained picked from the database. Eventual missing information about values and positions in the output file are obtained by searching utilities, and any information obtained through searching is stored in the database to supplement the existent information. This causes that subsequent interface calls use more and more direct file access, which speeds up interfacing.

The described interface enables the optimisation shell to work with the finite element environment Elfen, but it can be used as a model for building other file I/O interfaces. When deriving an interface for some other finite element simulation programme, portions related to formatting of the files should be modified, while much of the interface structure could be used with only minor modifications because organisation of input and output data would be similar in principle.

7. List of publications

a) Published

I. Grešovnik, T. Rodic, "Practical considerations regarding optimisation of shape in forming processes", V: M. Pietrzyk, Z. Mitura, J. Kaczmar (eds.), "*The 5th International ESAFORM Conference on Material Forming*", str. 27-30, Akadeimia Góeniczo-Hutnicza Kraków, 2002; ISBN 83-7108-098-0; Krakow, Poland, 14.-17. april. 2002.

I. Grešovnik, T. Rodič, S. Stupkiewicz, D. Cukjati, "*Application of Optimisation Techniques in Metal Forming", "34th Solid Mechanics Conference – volume of abstracts",* str. 109-110, Institute of Fundamental Technological Research, Polish Academy of Sciences (press: ATOS Poligrafia-Reklama), 2002; Solmech 2002, Zakopane, Poland, 2.-7. september 2002.

b) Submitted for publications

c) In preparation

I. Gresovnik and T. Rodic, "*An Integral Approach to Optimisation in Forming*

Technology", 5th World Congress of Structural and Multidisciplinary Optimisation, May 19-23, 2003 Lido di Jesolo, Italy